# 1.功能测试(func\_v0.02)

本文档主要对于测试功能和性能的说明。

实验流程应该是：独立执行单个六类指令测试 --> 接上soc\_sram\_func 并成功执行到64个测试点 --> 完成剩下的异常处理部分 --> 通过89个测试点 --> 接上AXI接口的Soc（完成sram到类sram的转接桥）-->完成cache上板测试性能。

## 1.1文件说明：

实验要求由实验指导书说明，本文档主要目的是辅助同学测试指令实现的正确性。测试的文件都在soft目录中，soft目录包括：func\_full, func\_part, funcTest\_independent,三个目录。

**func\_full**是所有的57条指令的总测试集，共89个测试点。由于我们的开发板的ram容量不够没有办法一次性的加载到FPGA上测试，所以后面我们拆成了三个部分也就是func\_part,此目录提供测试的源代码，调试的时候可以参考(func\_full/inst/xxx.S)文件对照。**Func\_part**目录包含三个obj文件：

obj\_1(对应funt\_full中的第1到47条测试)

obj\_2(对应funt\_full中的第48到64条测试)

obj\_3(对应funt\_full中的第65到89条测试)

前面两个obj不涉及异常指令和特权指令，只是基础的六类的指令的综合测试，具体的每条测试是什么可以自行参考(func\_ful/inst/)。使用此文档的测试需要街上sram接口的Soc才行。

**funcTest\_independent**是简单的六类指令的测试，可以提供在接上Soc之前的实现的测试。在自己设计的cpu上加载coe并对比测试就行。

对于每个阶段都提供一定的测试文件供同学们简单的判断指令执行的正确性。下面对于测试文件进行说明。

## 1.2 funcTest\_independent使用说明：

使用本测试集是基于自己设计的cpu之上的。在目录中有六个基础指令的测试集，将每个测试集的coe加载到指令存储器中仿真就可以测试。有几点说明如下：

需要将pc寄存器的初始值设置为00000000，依次将每个obj中的inst\_ram.coe文件加载到自己的指令rom中。执行仿真（提供一个简单的仿真测试文件test\_bench.v）。

在每个inst\_rom.S文件中都有对应的coe的源代码，一些汇编代码后面的注释为**正确执行到当前代码的时候，对应的寄存器的值，**所以对比仿真的波形图，将regfile模块中的32位寄存器添加入波形图中，对比相应的寄存器的值是否相同即可判断指令正确与否。

说明：在单独的测试中没有考虑到指令之间的数据相关，只可用作数据通路的检测。

## 1.3 Func\_part使用说明：

### 1.3.1 89 个功能点测试

测试环境依据 myCPU 的实现接口分为两个环境：SRAM 接口的 myCPU 使用 func\_test/soc\_sram\_func 目录下环境；AXI 接口的 myCPU 使用 soc\_axi\_func 下的 SoC\_AXI\_Lite 环境（暂时不考虑axi接口,,所以没有给出）。

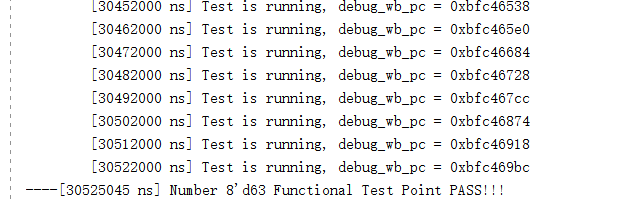
具体功能测试可参考doc/功能测试说明\_v0.01.sram接口的Soc的结构如下：



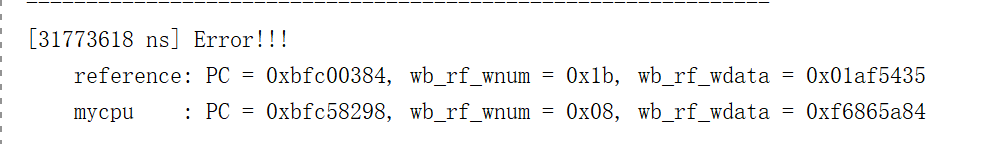
Ps由于现在取数据的来源有两种：data\_ram和confreg（外设模块），而我们的CPU得到的是虚拟地址，所以传出来的地址需要映射成物理地址。Soc将地址进行了一个分配，地址为1fafxxxx（物理地址）的时候将访问confreg。只需要在取数据的时候将地址判断一下，如果第31位为1，则需要将地址[31:29]三位数据置为0。（具体地址转换可参考mips32手册）

#### 1.3.1.1 测试结果判断

仿真结果正确判断很简单，就是看 Vivado 控制台打印 Error 还是 PASS，这是依据参考模型比较验证而打印的。如果仿真过程中打印了 Error，会同时打印出 myCPU 执行时的 debug 信息和参考的 debug 信号，随后可以很快的定位 bug，bug 就在打印 Error 的 PC 处，或者在其前几条指令处，这时就需要观察波形进行 debug 了。如图：



如果发生错误则会打印出正确pc写回的寄存器号，写回的数据的值，如图：



#### 1.3.1.2 Trace 比对机制使用

Trace 对比机制，对 soc\_sram\_func 和 soc\_axi\_func 都适用。如果需要使用该机制，需要 CPU 封装为相应的接口，将初始入口改为bfc00000，如下表展示了 SRAM 接口下对 myCPU 接口的实验要求，参考发布包 func\_test/soc\_sram\_func/rtl/soc\_lite\_top.v里对 myCPU 的调用。如果 CPU 实现为 AXI 接口，则将下表的取指/数据 SRAM 接口统一封装成 AXI 接口，参考发布包 func\_test/soc\_axi\_func/rtl/soc\_axi\_lite\_top.v 里对 myCPU 的调用。

关于 Trace 比对机制的详细使用参考发布包目录 doc/A11\_Trace 比对机制使用说明.pdf.



由于功能测试拆成了三个部分，我们的golden\_trace也生成了三个部分，所以在使用trace的时候需要做一些修改：如果使用obj\_1测试的时候，需要将testbench/mycpu\_tb.v中的

`define TRACE\_REF\_FILE "../../../../../../../cpu132\_gettrace/golden\_trace.txt"

此处最后面修改成

`define TRACE\_REF\_FILE "../../../../../../../cpu132\_gettrace/golden\_trace\_1.txt"

同理使用哪个obj，就需要将testbench文件修改成对应那一条golden\_trace，如果不行就将路径改为对应golden\_trace的绝对路径，注意斜杠。

# AXI接口功能测试（func\_v0.03）

## 2.1新增文件：

+ func\_test\_v0.03\cpu132\_gettrace\golden\_trace.txt

供同学仿真所有89功能测试的验证。

+ src\axi\_interface.v

axi接口转接桥，对接axi总线，cpu接口为类sram接口。

+ soc\_axi\_func

总线接口为axi接口。

+ doc\A12\_类SRAM接口说明.zip

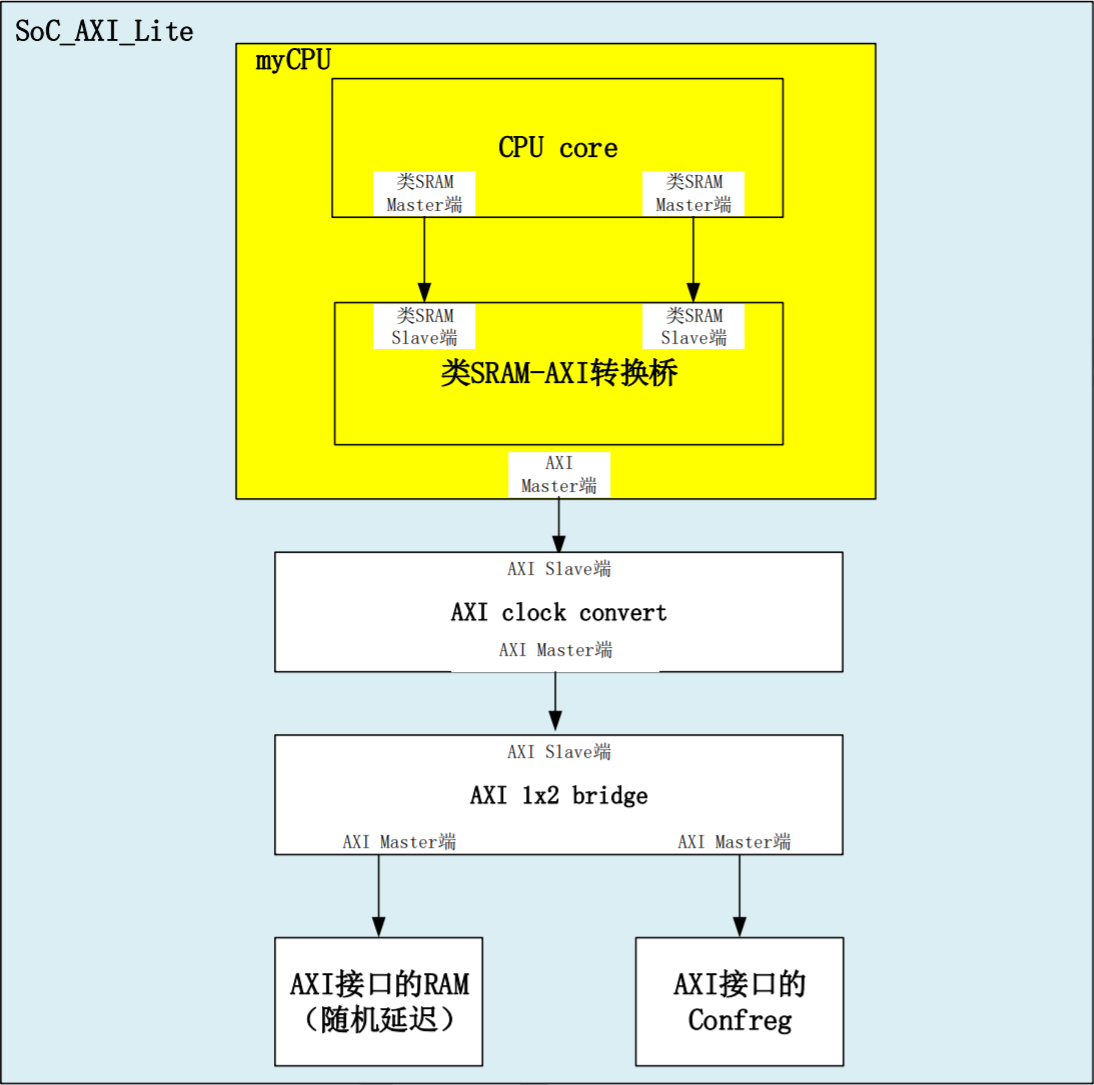
类sram接口的说明。

+ doc\AMBAaxi.pdf

AXI接口的说明

## 2.2 接口说明

此阶段，需要将之前的sram接口转换为AXI的接口，Soc由AXI接口连接，结构图如下：



与之前的sram的Soc不同，AXI接口的Soc将指令存储器和数据存储器集中在一个RAM之中。对于流水线CPU，如果访存阶段为store/load指令的时候就会和取指阶段发生冲突，因为现在的myCPU是通过AXI连接到总线的，而AXI只有一个地址线，所以我们需要解决这种情况下，先取指还是先取/存数据，这种冲突我们称之为**仲裁**。

仲裁我们在给出的转接桥里面已经解决了，然后会通过类sram接口的inst\_data\_ok,和data\_data\_ok告诉cpu有没有完成。

同学们需要做的事情就是将我们cpu的sram接口改为类sram接口：自己添加模块SRAMArbiter.v用作sram到类sram的转接。

## 2.3 功能测试的使用

同上个发布包的说明，但是存储器的ip核为u\_axi\_ram文件下的ram存储器。